BASIC INFORMATION

Name: Ashik Meerankutty

Major: Computer Science and Engineering,

University: College of Engineering Adoor

Github: @ashikmeerankutty

Website: https://mrdevops.co

Email: ashik9591@gmail.com

Phone: +91 9995736582

Postal Address: Shainy Manzil, Chottupara P.O, 685552, Kerala India

Elastic Auto Docs

AUTOMATIC DOCUMENTATION SYSTEM FOR TYPESCRIPT COMPONENTS

Abstract

A better automatic system for documenting EUI TypeScript components that extract the acceptable values for each prop on the API. This would be done through automatic transversal of the EUI components itself, combined with the type values from TS and inline comments.

Comparisons

The docs generated for `EuiToggle` toggle component by various tools

Existing

The tool used by Eui uses an old version of `babel-plugin-react-docgen: 3.1.0` that uses a lower version of `react-docgen: 4.1.0` that only supports javascript and flow and not typescript.
For supporting typescript by `react-docgen` a custom babel plugin is used to add PropType to typescript components. This method is good but a lot of manual addition has to be done for new types.
The latest version of `react-docgen` supports typescript and flow, but has many demerits.

Sample output: https://pastebin.com/tzd8rUtG

Demerits of latest version of `react-docgen`

- react-docgen will not be able to grab the type definition if the type is imported or declared in a different file.This is a major drawback as types such as `CommonProps` is used in almost all components.
  https://github.com/reactjs/react-docgen#flow-and-typescript-support
- In functional components we have to use `props:Type` format otherwise the parser doesn't work.

```
export const EuiToggle: React.SFC<EuiToggleProps> = ({
  id,
  className,
  checked,
  children,
  inputClassName,
  isDisabled,
  label,
  name,
  onChange,
  title,
  help,
  value,
  'data-test-subj': dataTestSubj,
  ...rest
}) => {
```

Has to be converted to

```
export const EuiToggle: React.SFC<EuiToggleProps> = (props: EuiToggleProps) => {
 const {
   id,
   className,
   checked,
   children,
   inputClassName,
   isDisabled,
   label,
   name,
   onChange,
   title,
   help,
   value,
   "data-test-subj": dataTestSubj,
   ...rest
 } = props;
```

Other similar libraries

Extract React Types: atlassian/extract-react-types

The `extract-react-types` package is used by atlaskit by Atlassian however the response from this is better than the existing one but it posses some major drawbacks such as in the case of EuiToggle it can't find the types of ToggleType hence it breaks and doesn't produce an output.

Sample output: https://pastebin.com/seGzrA3m

Babel React Docgen: storybook/babel-plugin-react-docgen.

This is the latest version of `babel-plugin-react-docgen` and it supports the latest version of `react-docgen` that supports typescript. The resulting `docgenInfo` for typescript components now contains a ` tsType` field as shown in the sample output that gives information about the typescript type and also it parses description information from the comments perfectly. As the Eui depends on the leading comment as the description of the prop. This tool is perfect for the descriptions. ` react-docgen ` will not be able to grab the type definition if the type is imported or declared in a different file. So a custom method has to be found to resolve this issue.

Sample output: https://pastebin.com/YJLp1XbQ

React Docgen Typescript: https://github.com/styleguidist/react-docgen-typescript

This is so far the best method I have found. The results are pretty satisfying. We could use a method somewhat similar to that of `react styleguidist` uses to fetch the data. We could extend this tool as per our needs. We might need to create a babel plugin to inject the docs info generated by this library to our components, details regarding this is explained in the next section.

Sample output with HTML props: https://pastebin.com/3K0utNS3
Sample output without HTML props: https://pastebin.com/y4ZgTrRN

While removing the HTML props we could whitelist some HTML props. Two methods can be used for the same:

1. By whitelisting the required props inside the plugin
   Eg:

   https://github.com/ashikmeerankutty/react-docgen-babel-plugin/blob/051297a50ffe950068055a9051fbc8828b6ebd3f/plugin/index.js#L33

   Here `className` is whitelisted and all other html props are omitted. Also in the same manner we could omit or include `fileNames`

2. By using `commonProps` as used by Eui.
   Eg:
   Here `EuiToastProps` interface extends `commonProps`

   https://github.com/ashikmeerankutty/react-docgen-babel-plugin/blob/051297a50ffe950068055a9051fbc8828b6ebd3f/src/components/toast/toast.tsx#L18

   The docs generated will have these `commonProps`
   https://ashikmeerankutty.github.io/react-docgen-babel-plugin/

   The only thing to note is the commonProps must come before HTML props while extending some interfaces or types.

Also we could mention if the component inherits props from a HTML element by showing if the component supports HTML attributes by adding components to an array if it supports HTML attributes and adding a boolean key `hasHtml` that equals true to docs info if the component is in the array and added this key to generated docsInfo.

This method will be similar to :

https://github.com/ashikmeerankutty/react-docgen-babel-plugin/blob/8873c1601793884e258c8c0a6c35c156b51c4e36/plugin/index.js#L21-L52

Proposed System

The proposed system consists of a new babel plugin that can add the generated docs for all `ts` | `tsx` files and inject the info to the component itself so that we can access that information right inside the react component itself.

- The docs info for the `ts` | `tsx` files will be generated with React Docgen Typescript parser.
- By default the parser supports exported types.

  For Eg:

  https://github.com/ashikmeerankutty/react-docgen-babel-plugin/blob/7c23629234155d510f5bfa6d4d3806f4899ddf67/src/components/toast/toast.tsx#L5

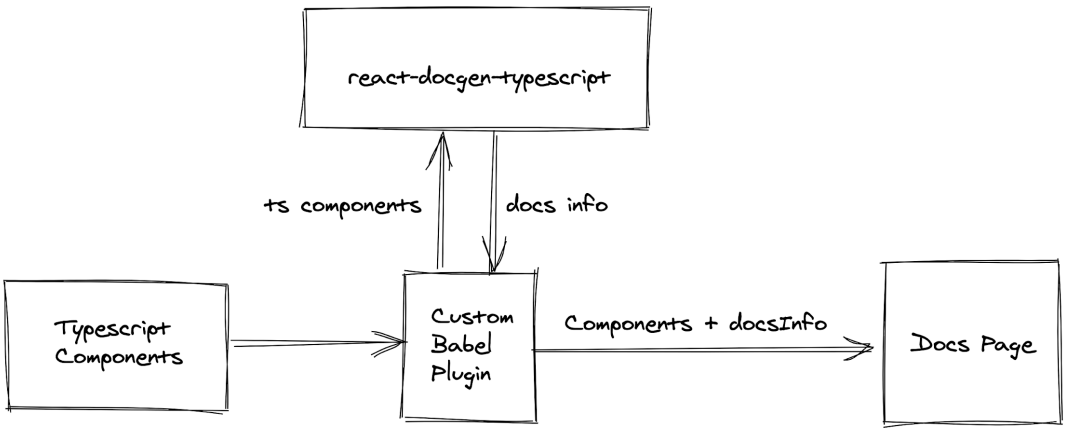
  Here `IconType` is exported from another file.

  The output of the parser can be seen here:

  https://ashikmeerankutty.github.io/react-docgen-babel-plugin/

  Under the `Toast Info` on the props `iconType` shows the exported props.

- We could then inject the generated docs info to our components with our new custom babel plugin, this will be similar to what [Babel React Docgen] plugin does.



Challenges

The docs info produced by [React Docgen Typescript] has the following demerits :

- If `strict: true` is enabled in tsconfig then an `undefined` is appended to all types that are not `required`.

  Before

data-test-subj	string | undefined
checked	boolean | undefined
isDisabled	boolean | undefined
name	string
help	enum ToggleType ["single", "multi"]
label	string
inputClassName	string | undefined
value	string | number | undefined

  Removing `strict: true`

data-test-subj	string
checked	boolean
isDisabled	boolean
name	string
help	enum ToggleType ["single", "multi"]
label	string
inputClassName	string
value	ReactText

When removing `strict: true` some types are not expanded.

Possible fixes:

- Parse the type string and remove the undefined from the end if the prop is not required.

Preview of Work

Generated props for EuiToggle and EuiToast

Plugin Repo: https://github.com/ashikmeerankutty/react-docgen-babel-plugin

Demo: https://ashikmeerankutty.github.io/react-docgen-babel-plugin/

Schedule of Deliverables

Phase 0: Community Bonding Period (May 4 – June 1)

- During the community bonding, the main focus will be to frame a roadmap for the project with the guidance of the mentor.
- This period will also be used to study the architecture of 'Elastic EUI' and to figure out a concrete plan for implementing the system using discussions.
- This time will be used to discuss and solve the challenges of the proposed system.
- Documenting the test cases that plugin should handle. The test cases will also include all the issues faced by the current eui docs system.

Phase 1: Coding Period 1 (June 1 – July 3)

Week 1 (June 1 - June 6)

- Development of the custom babel plugin which injects the docs info generated from the react-docgen-typescript to the `ts` and `tsx` components.
- Tests will be written for the test cases documented during the community bonding period.

Week 2 (June 7 - June 13)

- Continue development of the babel plugin.
- Initial version of the plugin will be ready by the end of this week.
- Initial testing will be done with some small components in eui.
- Documenting and fixing the bugs arose during the testing.

Week 3 (June 14 - June 20)

- Testing the plugin with some big components in the eui.
- Documenting and fixing the bugs arose during this testing.

Week 4 (June 21 - June 27)

- Fix all the remaining bugs.
- A production ready version of the plugin will be ready by this period.

First Evaluation Period

Phase 2: Coding Period 2 (July 3 – July 27)

Week 1 (July3 - July 11)

- Add the custom babel plugin to the eui docs.
- Tweaks that should be made for the new plugin to work in eui docs.
- Verify that the docs info is injected to all components.
- Fix any issues, if there is, injecting the docs info to components.

Week 2 (July12 - July 18)

- Make necessary changes to the Guide Section page to use new docs info.
- Verify if props generated by all the components are correct
- Fix any bugs.

Week 3 (July19 - July 25)

- Buffer week for any unexpected issues.

Second Evaluation Period

Phase 3: Coding Period 3 (July 31 – August 24)

Week 1 (July 31 - August 8)

- Convert the `src-docs` project to typescript.
- During the conversion the existing docs system will be removed as it is not necessary.

Week 2 (August 9 - August 15)

- Continue conversion of docs to typescript and removal of the old docs system.

Week 3 (August 16 - August 22)

- Test the whole project for bugs.
- Document all the changes made.

Week 4 (August 23 - August 29)

- Buffer week for any unexpected issues.

Final Evaluation Period

Phase 4: Final Evaluation (August 31)

All documentation, modules, and tests will be uploaded and make changes based upon the feedback of the mentor. All the deliverables promised for GSoC will be provided by this stage.

Availability

I have 4 university exams between May 18 and June 15. These exams may be postponed as a result of covid-19. Since the exam timetable is not published I can't say the exact days I will be unavailable. Exams usually take 2 weeks. So I won't be available for 2 weeks between May 18 and June 15. July and August I will be completely free.

Pull Requests to EUI

1. Added props to highlight all matches in EuiHighlight
2. Added prepend and append to EuiComboBox
3. Fixed data grid break on invalid schema
4. Added isSortable to Datagrid columns
5. Adding automated accessibility tests for layout pages
6. Typo fix in EuiDataGrid docs
7. Inherit commonly used data on relative tab on EuiSuperDatePicker
8. Inherit selected value to quick select in EuiSuperDatePicker
9. Added prepend and append support for EuiFieldPassword
10. EuiSuperDatePicker should show tooltip for the popover close event
11. Fixed EuiSuperDatePicker "End date" popover shows wrong relative time
12. Focus trap requires two clicks when focus is on EuiSelect
13. Fixed EuiSuperDatePicker input focus.

Pull Requests to Kibana

1. Create empty string filters when value not specified